Exercise 1

(10 points)

The goal in this year's exercise is to develop an XML-based format to represent data for an auction house (like eBay). In Exercise 1 you will create an XML Schema and XML document and in Exercise 2 you will be querying and transforming this document with XML-related technologies.

An auction house offers products. A user can place a bid on a specific product. After a set date, the auction for a product ends. The user with the highest bid wins the product.

Remark: The following data format can be seen as a database for an auction house.

In this exercise we will develop the XML Schema and an XML Document, that validates against the XML Schema. Afterwards, we will translate this XML Schema into a Document Type Definition (DTD).

XML Schema `auctions.xsd`

The first part of this exercise consists of writing an XML Schema document `auctions.xsd`. The XML Schema should allow for XML Documents according to the following definitions:

Element `auctions`

The root element `auctions` stores all the relevant information for the auction house.

It contains the following 3 child elements in exactly this order, but each element is **optional**:

- `products`
- `users`
- `bids`

Element `products`

The element `products` has no attributes and is a child element of `auctions`. The subtree rooted at `products` stores the products. It may contain an unbounded number of `product` elements.

Element `product` (child of `products`)

The element `product` stores a product. It contains in this order:

1. exactly one `name` element, which stores the name of the product as string,
2. an optional `description` element, describing the product,
3. an optional `expired` element, which is empty; and
4. zero or an unbounded number of `category` elements.

The `product` element has two attributes:

- an `id`, which is a nonnegative number and should globally identify this product; and
- an `auctionEnd`, which stores the date in the format "YYYY-MM-DD", on which the auction ends.

Element `description`

The element `description` is used to give a product description. It can contain elements `it` and `a`, and additionally, also text (in any order, with any multiplicity). The element `a` contains just text and has an attribute `href`, which contains a link to some other page. The element `it` contains text.

For example

```
<description>This is an <a href="example.htm">example</a> <it>text </it>!
</description>
```

is a valid `description` element.

Element `category`
The element `category` assigns a category to a product. The value of the element is either "book", "movie" or "music".

Element `users`
The element `users` has no attributes and is a child element of `auctions`. The subtree rooted at `users` saves the users. It may contain an unbounded number of `user` elements.

Element `user`
The element `user` describes a possible user of the auction house. It has the attribute `username`, which should be unique and identifies the user. Additionally, it has an attribute `password` storing the password of the user. The element `user` has the following child elements in exactly this order:

1.
   - Either the element `fullname` that contains a string, or
   - the element `name` which contains the elements `firstname` and `lastname` (both contain a string).
2. An unbounded number of `email` elements, all of which contain a string.
3. The element `balance` that contains a nonnegative integer.

Element `bids`
The element `bids` has no attributes, but may contain an unbounded number of `product` elements. These `product` elements are of different type as the child element `product` of `products`. This type is described next.

Element `product` (child of `bids`)
The element `product` describes the bids for this product. It has an attribute `id` which is a nonnegative integer and refers to the `id` of a `product` element in `products`. Each product may only have one `product` element as child of `bids`. It has one or an unbounded number of `bid` elements.

Element `bid`
The `bid` element stores a nonnegative integer, which represents the bid of a specific user. Its only attribute `user` references a `username` attribute of a `user` element.

Keys
Add the following keys to your document:

- `userKeys` for the usernames.
- `productKeys` for the products (child of `products`).

The keys are referenced in the following fields:

- `userKeys` is referenced in the `user` attribute of the `bid` element.
- `productKeys` is referenced in the `id` attribute of the `product` element (child of `bids`).

General Remarks
Please pay attention to the following remarks:

- If nothing else is mentioned all elements and attributes are required. The word "may" hints you to optional elements or attributes.
- All numbers are integers.

Summary

- **Files**: `auctions.xsd`
- **Maximum number of points**: 5

XML Document `auctions-xsd.xml`

Create an XML Document `auctions-xsd.xml` for the XML Schema `auctions.xsd`. The XML

Document should satisfy the following criteria:

- Create at least three `product` elements (child of `products`).
- Create three users.
- Create for each `product` at least one `bid` element.
- Create at least six bids.

Make sure that your XML Document `auctions-xsd.xml` validates against your XML Schema `auctions.xsd`. This can be done with the following command (after you have installed `xmllint`:

```
xmllint --schema auctions.xsd auctions-xsd.xml
```

Downloads and user instructions to `xmllint` can be found on our exercise page.

Summary

- **Files**: `auctions-xsd.xml`
- **Maximum number of points**: 1

Document Type Definition (DTD) `auctions.dtd`

Create a Document Type Definition (DTD) `auctions.dtd`, based on the above description. If some specification has a very complicated or no equivalent at all in DTDs, then make reasonable assumptions. If it is not possible that your XML document from above validates against your DTD, then create a new XML document `auctions-dtd.xml`, with the same information as in `auctions-xsd.xml`, that validates against your DTD.

Be prepared to explain in the assignment discussion which functionalities are not possible with DTDs. But, you don't need to create large number ranges as enumerations (e.g. "Numbers between 1 and 72 as an enumeration of 72 numbers).

Remarks

Make sure that at least the XML Document `auctions-dtd.xml` validates against your DTD `auctions.dtd`. This can be done with the following command (after you have installed `xmllint`:

```
xmllint --dtdvalid auctions.dtd auctions-dtd.xml
```

Summary

- **Files**: `auctions.dtd`, `auctions-dtd.xml`
- **Maximum number of points**: 4

Submission

In total you have to upload the following files:

- `auctions.xsd`
- `auctions-xsd.xml`
- `auctions.dtd`
- `auctions-dtd.xml`

These files should be zipped and the resulting ZIP-file `exercise1.zip` has to be uploaded until 05.05.2015 23:59 in TUWEL . We will grade the last uploaded solution.

Assignment discussion

You can receive at most 10 points for Exercise 1. During the assignment discussion we will not only check your solution for correctness, but will also ask some questions regarding the used technologies. Please be prepared to answer the questions mentioned in this document as well.

To obtain the full number of points your solution has to be solved correctly and you have to be able to explain it. Copied solutions are awarded 0 points!

In your own interest come **ontime** to your assignment discussion or else we don't guarantee that your solution is completely checked in the remaining time of the reserved slot.